УДК 004:056.53

УЯЗВИМОСТИ В СОВРЕМЕННОМ КЛИЕНТ-СЕРВЕРНОМ ОБЩЕНИИ

О.И. РАЧИЦКИЙ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Описаны основные виды атак на веб-приложения, которые используют уязвимости протоколов сообщения между клиентами и серверами, или недочеты архитектуры некоторых приложений, дизайн которых был создан без учета уязвимостей.

Современные браузеры являются достаточно защищенными от практически любых действий злоумышленников, ставящих своей целью похитить данные пользователей или воспользоваться пользовательскими полномочиями. Но в силу особенностей клиент-серверного взаимодействия структуры запросов между клиентом и сервером и некоторых особенностей JavaScript, используемого как язык работы с сайтом с клиентской стороны, существуют типы атак, которые невозможно предотвратить только силами браузера. О них и пойдет речь в этой работе.

Наиболее популярная и простая атака – это CSRF (cross-request forgery) атака. При CSRF атаке злоумышленники заставляют клиента выполнить нежелательные действия в приложениях, в которых клиент аутентифицирован. Этот тип атаки основан на недостатках протокола HTTP. Алгоритм действия злоумышленников в данном случае простой:

- найти на целевом сайте GET или POST запрос, выполнение которого клиентом может вызвать интерес со стороны злоумышленников. Это может быть запрос на денежный перевод или передачу личных данных;
- сформировать запрос таким образом, чтобы целевой сайт отослал данные или осуществил денежный перевод в том виде, в котором это выгодно мошенникам;
- создать собственный вредоносный сайт и поместить туда скрипт, который автоматически будет или осуществлять перенаправление на нужный сайт (для GET запросов), или активировать отправление формы с нужными параметрами (для POST запросов);
- дать ссылку в каком-либо виде пользователю или заставить пользователя перейти по ссылке, используя вредоносный код или скрипт, встроенный каким-либо образом в брузер клиента;
- как только пользователь перейдет на вредоносный сайт, на этот сайт не будут отправлены его куки, созданные на целевом сайте, однако как только вредоносный скрипт осуществит запрос к целевому сайту от лица пользователя, в таком запросе браузер автоматически передаст все куки домена целевого браузера, и если целью злоумышленников было проведение денежной транзакции, то целевой сайт проверит достоверность куки пользователя, а после авторизует и обработает сформированный злоумышленниками запрос. Это является основной уязвимостью при хранении аутентификационной и авторизационной информации в куки.

Вторым популярным типом атаки является *XSS* (Cross-Site Scripting). Суть XSS заключается в том, что разнообразными способами на страницу сайта, отображаемую у клиента, внедряется вредоносный код. Причем не всегда для этого требуется взлом самого сервера. XSS атаки можно разбить на два подтипа: отраженные и хранимые.

Отраженные XSS атаки основаны на отраженной уязвимости, на сегодняшний день является самой распространенной XSS-атакой. Эти уязвимости появляются, когда данные, предоставленные веб-клиентом, чаще всего в параметрах HTTP-запроса или в форме HTML исполняются непосредственно серверными скриптами для синтаксического анализа и отображения страницы результатов для этого клиента, без надлежащей обработки. Отраженная XSS-атака срабатывает, когда пользователь переходит по специально подготовленной ссылке, в которой в параметрах запроса передается вредоносный JavaScript код, не кодируеющийся сервером и может быть исполнен в браузере клиента на странице, которую ему вернет сам сервер.

Отраженные атаки, как правило, рассылаются по электронной почте или размещаются на Web-странице. URL приманки не вызывает подозрения, указывая на надежный сайт, но содержит вектор XSS. Если доверенный сайт уязвим к вектору XSS, то переход по ссылке может привести к тому, что браузер жертвы начнет выполнять встроенный скрипт.

Хранимые XSS атаки являются наиболее разрушительными, потому что они возможны только тогда, когда злоумышленнику удается внедрить на сервер вредоносный код, выполняющийся в браузере каждый раз при обращении к оригинальной странице. Классическим примером этой уязвимости являют-

ся форумы, блоги, чаты, на которых разрешено оставлять комментарии в HTML формате без ограничений. Также эта уязвимость имеет место быть, когда на сервере разрешается хранить пользовательские тексты и рисунки. Скрипты вставляются в эти тексты и рисунки.

Еще одной распространенной ранее, но возможной даже в современном мире веб-технологий, является *SQL-инъекция*. Внедрение SQL в зависимости от типа используемой СУБД и условий внедрения может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.

Некоторые сервера опираются на клиентскую часть в плане составления запросов и не проверяют передаваемые параметры. Это может быть уязвимостью при внедрении в параметры SQL-кода, который будет исполнен сервером.

Также даже при защите от банального внедрения SQL-кода разработчики часто забывают о наличии оператора UNION в большинстве SQL СУБД, который призывает сервер выполнить помимо основной команды еще и дополнительную, описанную после этого оператора. Если говорить другими словами, язык SQL позволяет объединять результаты нескольких запросов при помощи оператора UNION. Это предоставляет злоумышленнику возможность получить несанкционированный доступ к данным.

Зачастую SQL-запрос, подверженный данной уязвимости, имеет структуру, усложняющую или препятствующую использованию UNION. В таких случаях злоумышленниками используется метод экранирования части запроса при помощи символов комментария в зависимости от типа СУБД. Для разделения команд в языке SQL используется символ *точка с запятой*, внедряя этот символ в запрос, злоумышленник получает возможность выполнить несколько команд в одном запросе, однако не все диалекты SQL поддерживают такую возможность.

Поиск скриптов, уязвимых для атаки, осуществляется злоумышленниками посредством изучения поведения скриптов сервера при манипуляции входными параметрами с целью обнаружения их аномального поведения. Манипуляция происходит всеми возможными параметрами: данными, передаваемыми через методы POST и GET, значениями куки, HTTP_REFERER (для скриптов), AUTH_USER и AUTH_PASSWORD (при использовании аутентификации). Как правило, манипуляция сводится к подстановке в параметры символа одинарной (реже двойной или обратной) кавычки.

Аномальным поведением считается любое поведение, при котором страницы, получаемые до и после подстановки кавычек, различаются (и при этом не выведена страница о неверном формате параметров). Наиболее частые примеры аномального поведения – выводится сообщение о различных ошибках, при запросе данных запрашиваемые данные не выводятся вообще, хотя страница отображается. Следует учитывать, что известны случаи, когда сообщения об ошибках в силу специфики разметки страницы не видны в браузере, хотя и присутствуют в ее HTML-коде.

Заключение

Несмотря на известность атак и общую осведомленность об их существовании избавиться от них невозможно из-за особенностей стандартов HTTP и модели серверов. Единственным вариантом защиты от такого типа атак является их предусмотрение и контроллирование необходимых для проведения атак условий на стороне сервера. Об этом следует помнить разработчикам сайтов.

ЛИТЕРАТУРА

1. Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), SQL Injection, HTML Injection, Etc [Электронный ресурс]. – Режим доступа: https://www.zscaler.com/blogs/research/cross-site-scripting-xss-cross-site-request-forgery-csrf-sql-injection-html-injection-etc . – Дата доступа: 26.09.2017.
2. CSRF and XSS: A Lethal Combination – Part I [Электронный ресурс]. – Режим доступа: http://resources.infosecinstitute.com/csrf-xss-lethal-combination/. – Дата доступа: 26.09.2017.
3. Understanding and selecting authentication methods [Электронный ресурс]. – Режим доступа: http://www.techrepublic.com/article/understanding-and-selecting-authentication-methods/. – Дата доступа: 26.09.2017.
4. XSS глазами злоумышленника [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/66057. – Дата доступа: 26.09.2017.